THREE PLOYS FOR ROBUST CO-GENERATION WITH GENERATIVE ADVERSARIAL NETS

BY

KRISHNA HARSHA REDDY KOTHAPALLI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Alexander Schwing

ABSTRACT

Generative adversarial nets (GANs) and variational auto-encoders enable accurate modeling of high-dimensional data distributions by forward-propagating a sample drawn from a latent space. However, an often overlooked shortcoming is their inability to find an arbitrary marginal distribution, which is useful for completion of missing data in tasks like super-resolution, image inpainting, *etc*., where we don't know the missing part ahead of time. To address such applications it seems intuitive at first to search for *that latent space sample* which 'best' matches the observations. However, irrespective of the GAN loss, unexpected challenges arise: we find that the energy landscape of well trained generators is extremely hard to optimize, exhibiting 'folds' that are very hard to overcome. To address this issue, in this thesis, three ploys are proposed which help to address the challenge for all investigated GAN losses and which yield more accurate reconstructions, quantitatively and qualitatively.

To my parents, for their constant love and support.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Alexander Schwing, who's constant help and motivation kept me going. I would also like to thank Xiaoyang Bai for coming up with great ideas and helping me in most of my experiments. This work would not be possible without both of them.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 RELATED WORK AND BACKGROUND 3

CHAPTER 3 ROBUST CO-GENERATION 6
3.1 Co-Generation Formulation and Challenges 6
3.2 Robust Approaches for Co-Generation 12

CHAPTER 4 EXPERIMENTS 15
4.1 Synthetic Data 15
4.2 Real Data 20

CHAPTER 5 CONCLUSION 24

APPENDIX A APPENDIX 25
A.1 Additional Analysis 25
A.2 Additional Qualitative Results 25

REFERENCES 33

CHAPTER 1: INTRODUCTION

Learning a joint distribution over a domain which subsumes multiple variables is useful across fields, from computer vision, and natural language processing to medical imaging, where it has been used, among others, for domain transfer [1, 2], inpainting [3, 4], image-to-image translation [5, 6, 7, 8, 9, 10, 11, 12], machine translation [13] and health care [14].

Classical approaches for learning of a joint probability distribution over a domain that is composed out of multiple discrete random variables are often collectively referred to as structured prediction [15, 16, 17]. Over continuous domains, among others, Gaussian Markov Random fields [18] have been used. Those joint probability distributions capture the likelihood of a configuration of the random variables which comprise the output space. Importantly, in many cases, it is computationally extremely demanding to find the most likely configuration even if we are given a labeling for a subset of the output space variables [19, 20].

To address modeling of high-dimensional distributions, generative adversarial nets (GANs) [21] and variational auto-encoders [22] evolved as compelling tools because of the underlying manifold assumption: a latent 'perturbation' is drawn from a simple distribution which is subsequently transformed via a deep net (generator) to the output space. While this permits easy sampling from the entire output space domain, it remains an open question how to sample from part of the domain given the remainder?

This question has been addressed in numerous works. For instance, for image-to-image translation [5, 6, 7, 8, 9, 10, 11, 12], mappings between domains are learned directly via an encoder-decoder structure. While such a formulation is convenient if we have two clearly separate domains, this mechanism isn't scaleable if the number of output space partitionings grows, *e.g.*, for image inpainting where missing regions are only specified at test time.

More formally, assume we are given at test time a partitioning of the output space $\mathcal{X}$ into two domains, the exposed and hidden part X_e and X_h. Co-generation (for conditional generation) involves drawing samples from $P(X_e = \boldsymbol{x}_e, X_h)$ given a test-time partitioning and an instantiation $\boldsymbol{x}_e$. This means co-generation is naturally formulated as a search for latent variable values given the observation, aided by backpropagation through the generator. Note that this procedure offers a powerful alternative to traditional inference methods because one does not need to build conditional models. While this is a compellingly general inference procedure, applicable even when both X_e and X_h are high dimensional, it is computationally challenging. While this has been observed before [4, 23], here we explain why.

In this paper, we find that, as the generator improves during training of a GAN, the search for a latent variable corresponding to $X_e = \boldsymbol{x}_e$ becomes harder. Successful training of the GAN

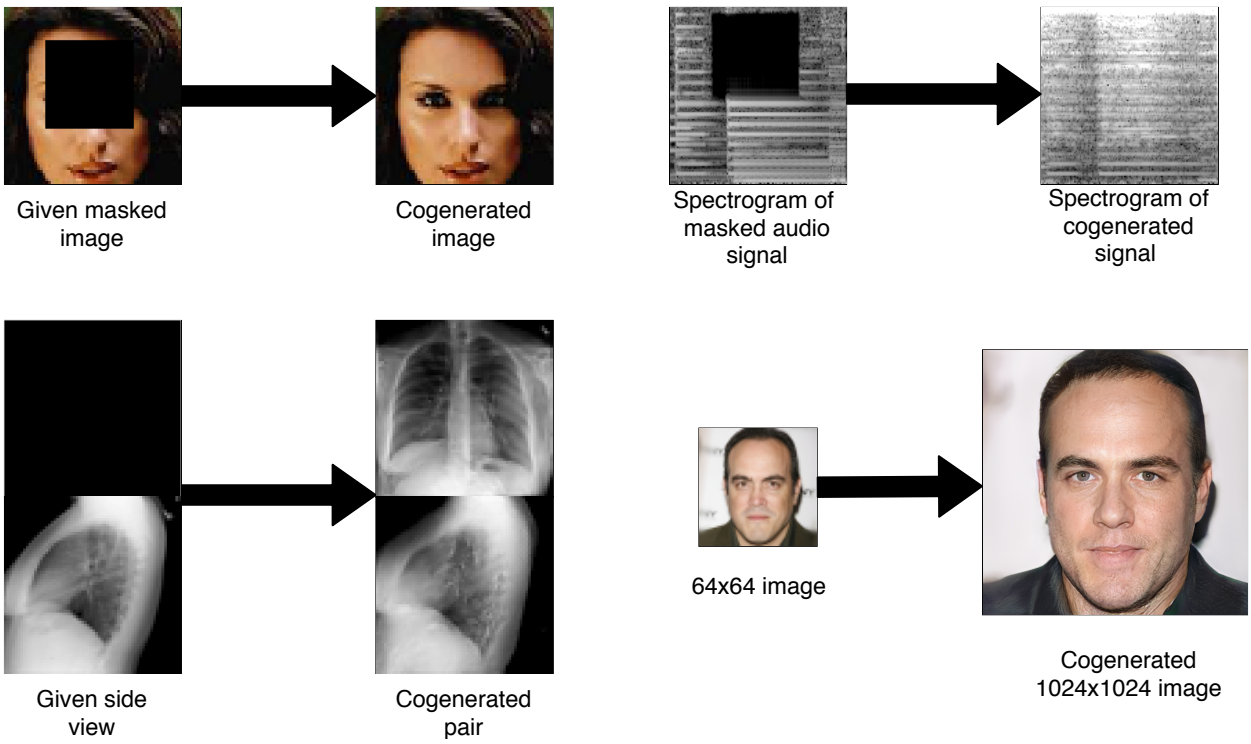


Figure 1.1: Some applications of the method

leads to an increasingly ragged energy landscape, making search for an appropriate latent variable via backpropagation through the generator harder and harder until it eventually fails. Quadratic algorithms don't help (*cf*. LBFGS in [23]), because the problem is due to the global structure of the landscape. We show three ploys that help significantly with this challenge, and demonstrate the efficacy of the proposed approaches on one synthetic and three real-world datasets. As illustrated in Fig. 1.1, our method can be used for various applications.

CHAPTER 2: RELATED WORK AND BACKGROUND

In the following we briefly discuss generative adversarial nets before providing background on co-generation with adversarial nets.

Generative adversarial nets (GANs) [24] have originally been proposed as a non-cooperative two-player game, pitting a generator against a discriminator. The discriminator is tasked to tell apart real images from those produced by the generator, while the generator is asked to make differentiation for the discriminator as hard as possible. For a dataset of samples $\boldsymbol{x} \in \mathcal{X}$ and random perturbations $\boldsymbol{z} \in \mathcal{Z}$ drawn from a simple distribution, this intuitive formulation results in the saddle-point objective

$$\max_{\theta} \min_{w} -\mathbb{E}_{\boldsymbol{x}}[\ln D_w(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}}[\ln(1 - D_w(G_\theta(\boldsymbol{z})))], \tag{2.1}$$

where G_θ denotes the generator parameterized by θ and D_w refers to the discriminator assessing the probability of its input argument being real data. We let $\mathcal{X}$ and $\mathcal{Z}$ denote the output space and the latent space. Subsequently, we refer to this formulation as the 'Vanilla GAN,' and note that its loss is related to the Jenson-Shannon divergence. Many other divergences and distances have been proposed recently [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36], some theoretically founded and others empirically motivated, to improve the stability of the saddle-point objective optimization during training and to address mode-collapse.

A frequently used GAN objective is the Wasserstein GAN with gradient penalty (subsequently referred to as 'WGAN-GP') [37], which addresses

$$\begin{aligned} \max_{\theta} \min_{w} \quad & \mathbb{E}_{\boldsymbol{z}}[D_w(G_\theta(\boldsymbol{z}))] - \mathbb{E}_{\boldsymbol{x}}[D_w(\boldsymbol{x})] \\ & + \lambda \mathbb{E}_{\hat{\boldsymbol{x}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2], \end{aligned} \tag{2.2}$$

where D_w refers to the discriminator parameterized by w and $\hat{\boldsymbol{x}} = \epsilon \boldsymbol{x} + (1-\epsilon) G_\theta(\boldsymbol{z})$ for ϵ a random number drawn uniformly from $[0, 1]$.

We find standard WGAN-GP to train more effectively when using $\log(\sigma(D_w(G_\theta(\boldsymbol{z}))))$ and $\log(\sigma(D_w(\boldsymbol{x})))$, improving the performance of WGAN-GP. Also since part of the optimization objective in co-generation involves the probability that a sample is true, this modification proves to be convenient for co-generation. Subsequently, D and D_w refer to the log-sigmoid of D_w's output when we talk about WGAN-GP.

Another recently proposed GAN objective is based on the sliced Wasserstein distance [29].

Different from the classical Wasserstein variants, [29] directly addresses the primal objective

$$\min_{\theta} \min_{M \in \mathcal{M}} \sum_{i,j} M_{i,j} \|\boldsymbol{x}_i - G_\theta(\boldsymbol{z}_j)\|_2^2, \tag{2.3}$$

where $\mathcal{M}$ is the set of all permutation matrices and $\boldsymbol{x}_i$ and $\boldsymbol{z}_j$ are the i-th and j-th sample respectively. We subsequently refer to their discriminator-augmented formulation as 'SWGAN,' possibly augmented by a gradient penalty term ('SWGAN-GP').

Co-generation, is *the task of obtaining a sample for a subset of the output space domain, given as input the remainder of the output space domain*. This task is useful for applications like image inpainting [3, 4] or image-to-image translation [5, 6, 7, 8, 9, 10, 11, 12]. Many formulations for co-generation have been considered in the past. However, few meet the criteria that *any* given subset of the output space could be used to generate the remainder.

Conditional GANs [38] have been used to generate output space objects based on a given input signal [39]. The output space object is typically generated as a whole and, to the best of our knowledge, no decomposition into multiple subsets is considered.

Co-generation is related to multi-modal Boltzmann machines [40, 41], which learn a shared representation for video and audio [41] or image and text [40]. Restricted Boltzmann Machine based encoder-decoder architectures are used to reconstruct either video/audio or image/text given one of the representations. Co-generation is also related to deep net based joint embedding space learning [42]. Specifically, a joint embedding of images and text into a single vector space is demonstrated using deep net encoders. After performing vector operations in the embedding space, a new sentence can be constructed using a decoder. No decoder is trained for generating images although this has been recently shown as well, as discussed next. Co-generation is also related to cross-domain image generation [43, 44, 45, 1]. Those techniques use an encoder-decoder style deep net to transform rotation of faces, to learn transfer of style properties like rotation and translation to other objects, or to encode class, view and transformation parameters into images.

Image-to-image translation is related in that a transformation between two domains is learned either via an Image Transformation Net or an Encoder-Decoder architecture. Early works in this direction tackled supervised image-to-image translation [46, 5, 47, 48, 49, 2] followed by unsupervised variants [50, 51, 10, 9, 52, 53, 54, 55]. Cycle-consistency was discovered as a convenient regularization mechanism in [56, 11, 8, 57] and a distance preserving regularization was shown in [58]. Disentangling of image representations was investigated recently [7, 6] and ambiguity in the task was considered by Zhu *et al*. [12]. Other losses such as a 'triangle formulation' have been investigated in [59, 60]. Attribute transfer [61], analogy learning [62, 44] and many style transfer approaches [63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77] just like feature learning via

inpainting [3] are also using an encoder-decoder formulation, which maps entire samples from one domain to entire samples in another domain.

Co-generation is at least challenging if not impossible for all the aforementioned works since decoders need to be trained for every subset of the output space domain. This is not scalable unless we know ahead of time the few distinct subsets of interest, which is the case for many of the aforementioned works, *e.g*., an entire image or an entire sentence is reconstructed.

Hence, to generate arbitrary sub-spaces, other techniques need to be considered. Some applicable exceptions from the encoder-decoder style training are work on style transfer by Gatys *et al*. [78], work on image inpainting by Yeh *et al*. [4], and coupled generative adversarial nets (CoGANs) by Liu and Tuzel [23]. In all three formulations, a loss which matches observations to generated data is iteratively optimized w.r.t. a latent space sample using back-propagation through the generator deep net. Note that generality of the output space decomposition obtained by iteratively back-propagating to the encoding space comes at the price of computational complexity, *i.e*., iterative back-propagation is obviously more expensive than a single forward pass through an encoder deep net.

In particular, Liu and Tuzel [23] learn a joint distribution over multiple domains by coupling multiple generators and possibly discriminators via weight-sharing. This is similar to our formulation. However, in contrast to Liu and Tuzel [23], we are interested in co-generation, *i.e*., the task of obtaining a sample for a subset of the output space domain, given as input the remainder of the output space domain. Liu and Tuzel [23] briefly discuss this topic when talking about "cross-domain image transformation," report to observe coverage issues and state that they leave a detailed study to "future work." In the following, we perform such an investigation.

CHAPTER 3: ROBUST CO-GENERATION

To this end, in Sec. 3.1 we first formalize co-generation in general and discuss the arising challenge, and finally, in Sec. 3.2, we provide ways to alleviate the issue.

3.1 CO-GENERATION FORMULATION AND CHALLENGES

Co-Generation Formulation: Given a trained generator G (for readability we won't make its parameters explicit from here on), which maps an input perturbation $\boldsymbol{z} \in \mathcal{Z}$ to the output space object $\tilde{\boldsymbol{x}} = G(\boldsymbol{z}) \in \mathcal{X}$, and given an arbitrary partition of an output space sample $\boldsymbol{x}$ into the exposed portion $\boldsymbol{x}_e$ and the hidden portion $\boldsymbol{x}_h$, *i.e.*, $\boldsymbol{x} = (\boldsymbol{x}_e, \boldsymbol{x}_h)$, we want to recover $\boldsymbol{x}_h$ from $\boldsymbol{x}_e$. Importantly, we emphasize that the partitioning of a sample $\boldsymbol{x}$ into exposed and hidden part is not known ahead of time and can vary from sample to sample; also, the exposed and hidden portions need not be spatially related, e.g., when they are different views of the same object or different subspaces of the frequency domain of a particular type of sound.

Recalling the manifold assumption, intuitively, we want to take advantage of the given generator G and find a perturbation $\boldsymbol{z}^*$ which accurately mimics the exposed portion, *i.e.*, the restriction of the generator on the exposed part $G(\boldsymbol{z}^*)_e \approx \boldsymbol{x}_e$. In this process we recover an estimate of the hidden part $G(\boldsymbol{z}^*)_h$. Consequently, to address this task, we want to find $\boldsymbol{z}^*$ such that (as illustrated in Fig. 3.1)

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} R(G(\boldsymbol{z})_e, \boldsymbol{x}_e) + \alpha \cdot L(D(G(\boldsymbol{z}))), \tag{3.1}$$

where $\alpha \geq 0$ is a weight, R, the reconstruction loss, is a quality assessment function such as mean squared error (MSE) or mean structured similarity (MSSIM) [79], and L, the likelihood loss, is a function assessing realism of the generated sample $G(\boldsymbol{z})$ by leveraging the discriminator output D. For instance, the negative log-likelihood $L = -\log(D_w(G(\boldsymbol{z}))$ or the log-D-trick version $L = \log(1 - D_w(G(\boldsymbol{z})))$ can be used.

The performance of co-generation is then evaluated by the overall reconstruction error $\hat{R}$, *i.e.*, via

$$\hat{R}(\boldsymbol{z}^*, \boldsymbol{x}) = R(G(\boldsymbol{z}^*), \boldsymbol{x}). \tag{3.2}$$

The objective in Eq. (3.1) looks very unassuming. Consequently, the general approach for co-generation has been to (1) sample a $\boldsymbol{z}$ randomly, to (2) backpropagate the loss $R + \alpha L$ through the generator G to obtain a gradient w.r.t. $\boldsymbol{z}$, and to (3) update the perturbation $\boldsymbol{z}$ using a gradient update. More complex techniques such as LBFGS have been used as well. However, as we will show next, this straightforward gradient descent-based approach faces some unexpected

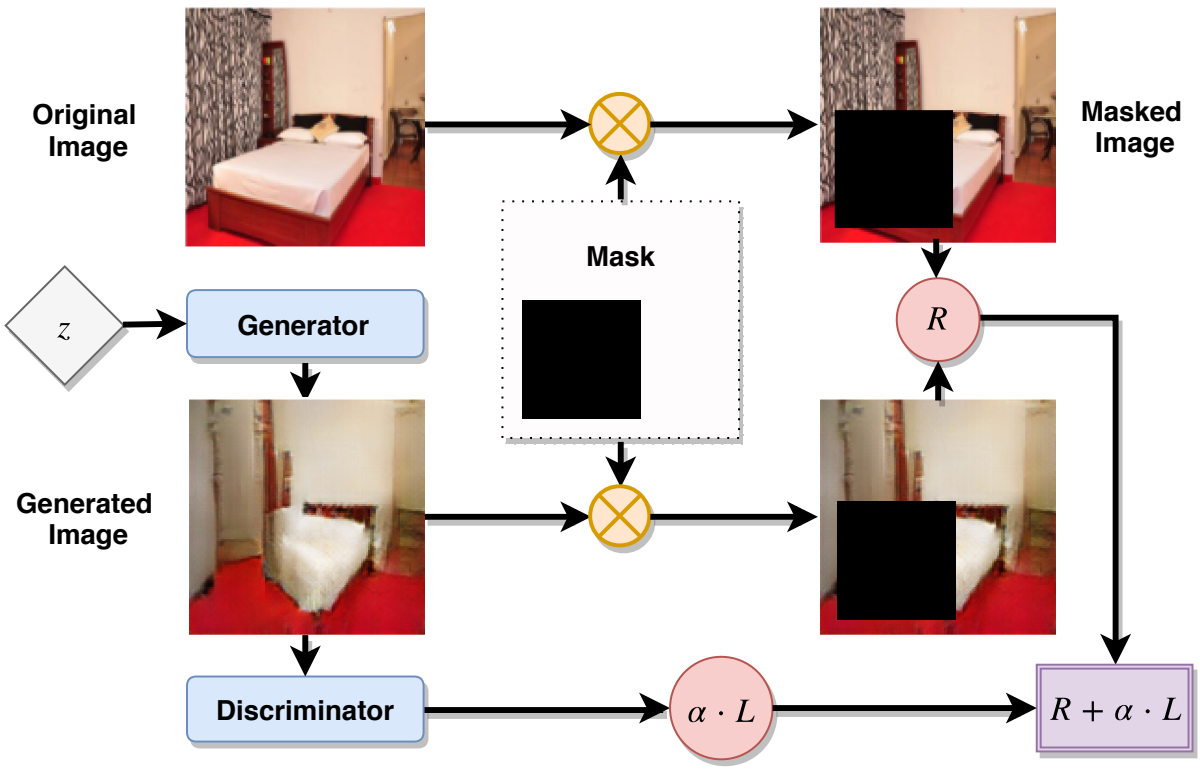


Figure 3.1: Illustration of co-generation.

challenges, making it hard to converge in reality, particularly if the generator is well trained.

Challenge: We found the aforementioned gradient descent procedure w.r.t. z to yield inaccurate and unstable results, which seems counterintuitive. To obtain more insights, on synthetic and real data, for which we provide details later, and across the GAN variants discussed in Sec. 2, we illustrate in Fig. 3.2 the reconstruction error $\hat{R}$ (MSE in our case) over the iterations of GAN training. Common to many results on multiple datasets and across GAN training algorithms: the reconstruction error is often better early in GAN training (low iteration numbers) and worsens eventually (high iteration numbers). This observation can be explained intuitively if generator G and/or discriminator D training make the optimization landscape for the objective given in Eq. (3.1) increasingly rough.

To validate this hypothesis on synthetic data with two-dimensional output and perturbation space, we illustrate the objective given in Eq. (3.1) for one exposed dimension over both $\mathcal{X}$ and $\mathcal{Z}$ space across some GAN training iterations in the Fig. 3.4 and Fig. 3.3 respectively. With GAN training progressing, we find an increasingly better fit of the modeled distribution. But importantly, gradient descent in the $\mathcal{Z}$-space, illustrated in Fig. 3.3, is eventually a formidable undertaking.

In Fig. 3.5 we show for different GAN losses (a)-(d), (1) samples from the synthetic data distribution (red points), (2) samples from the distribution induced by the generator in $\mathcal{X}$-space (black points), and (3) contours corresponding to the discriminator log-likelihood. We observe all GAN losses to accurately recover the synthetic data distribution for both the discriminator and the gen-

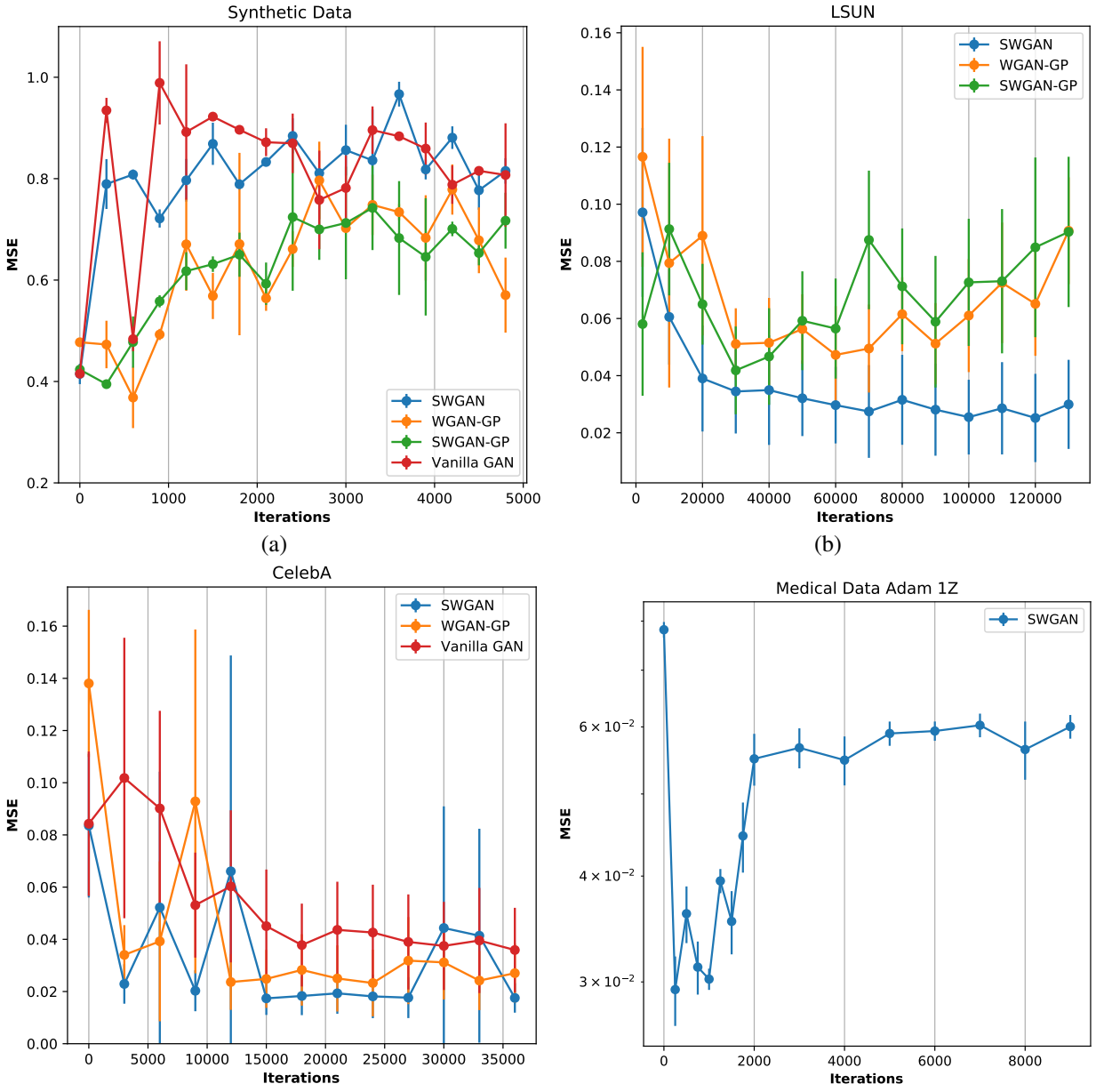


Figure 3.2: Reconstruction error over the number of GAN training iterations for different datasets ((a) synthetic; (b) LSUN, (c) CelebA, (d) Medical) and GAN losses (see legends within each panel). Observe: better trained GANs often yield worse reconstruction error.

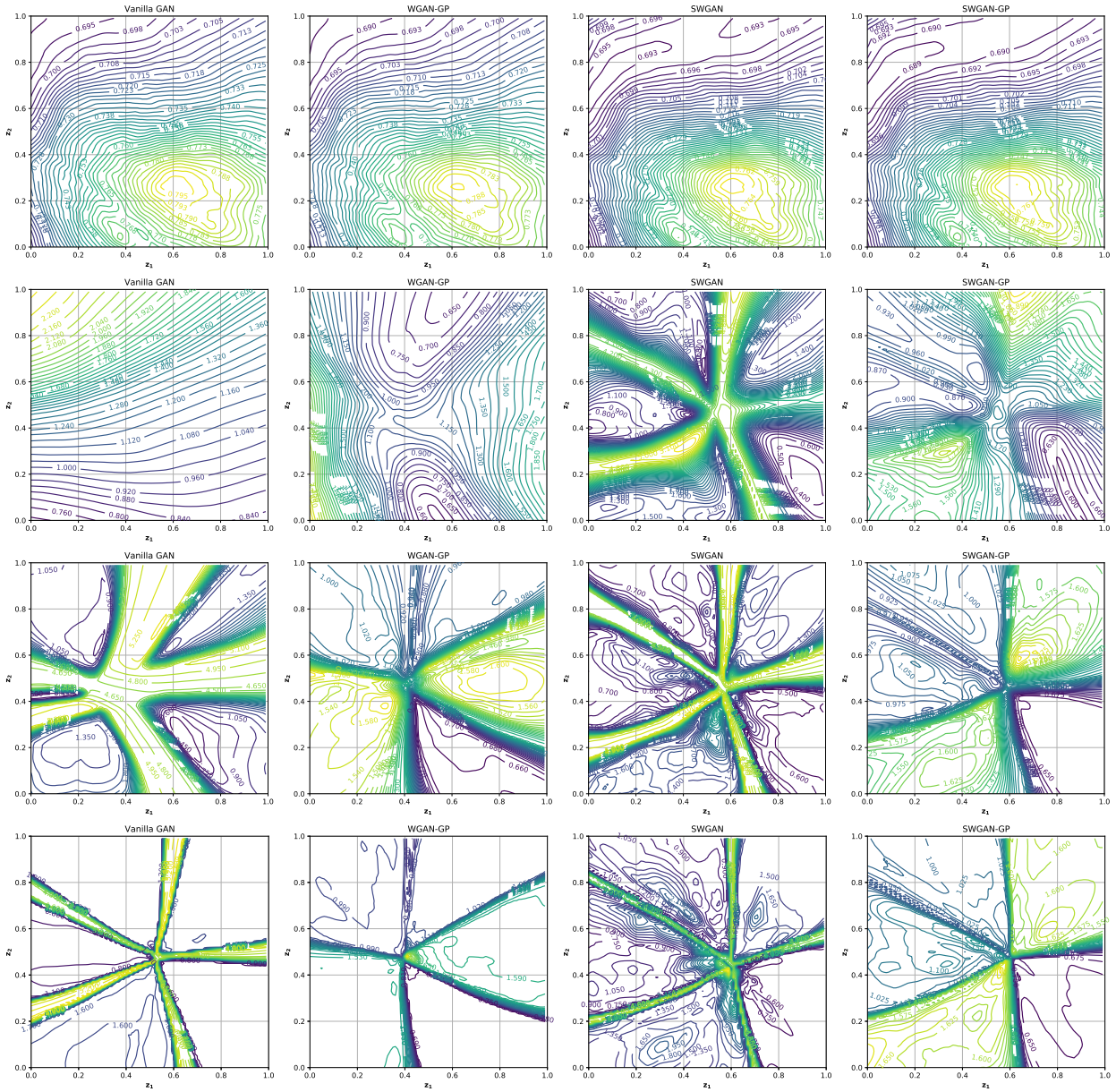


Figure 3.3: Loss in $\mathcal{Z}$-space: (a) Vanilla, (b) WGAN-GP, (c) SWGAN, (d) SWGAN-GP at 0, 300, 1200 and 18000 iterations (top to bottom).

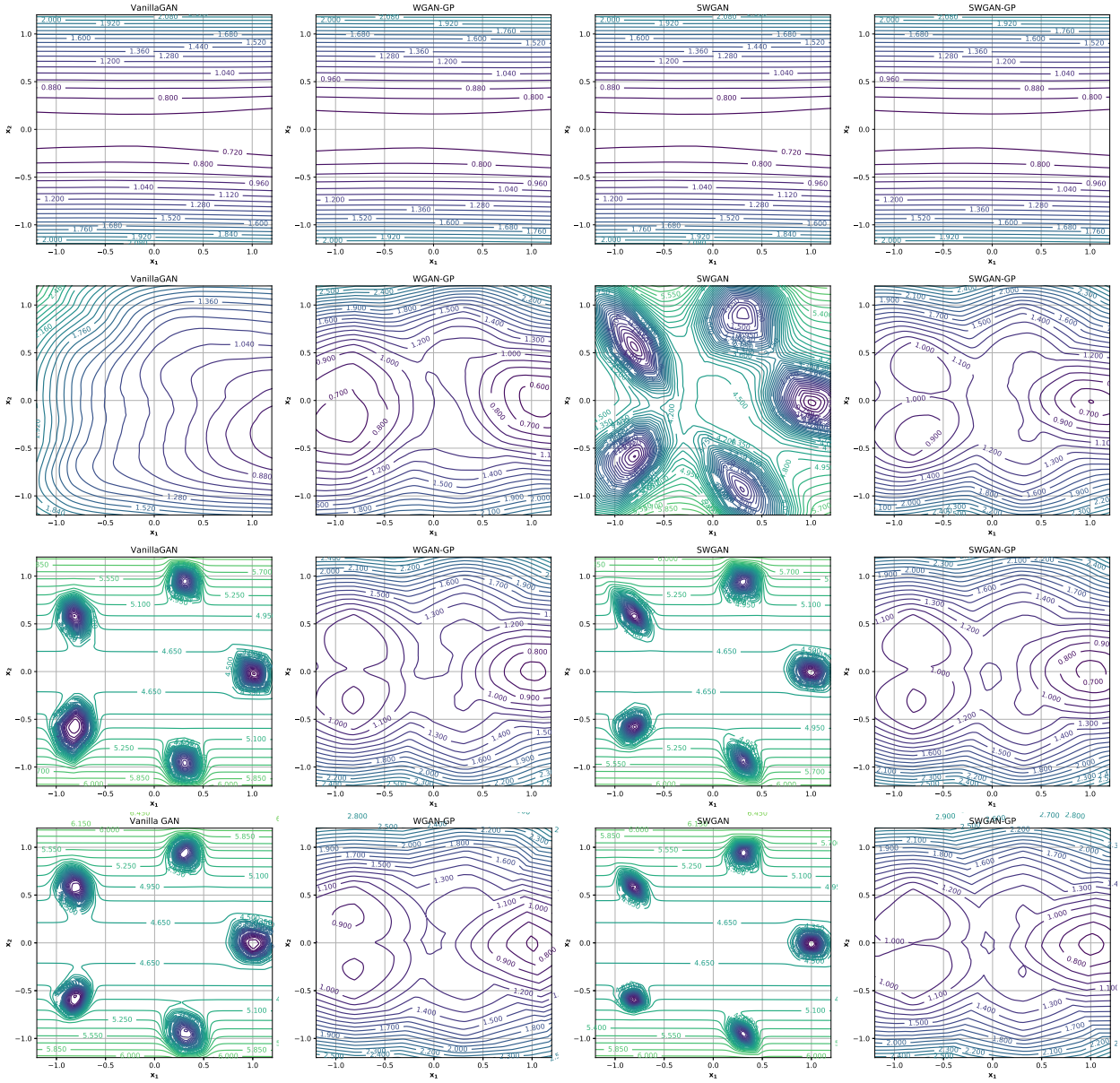
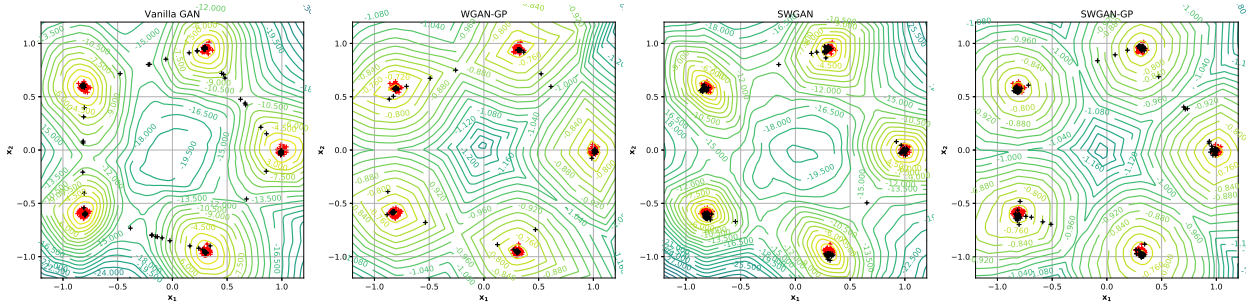
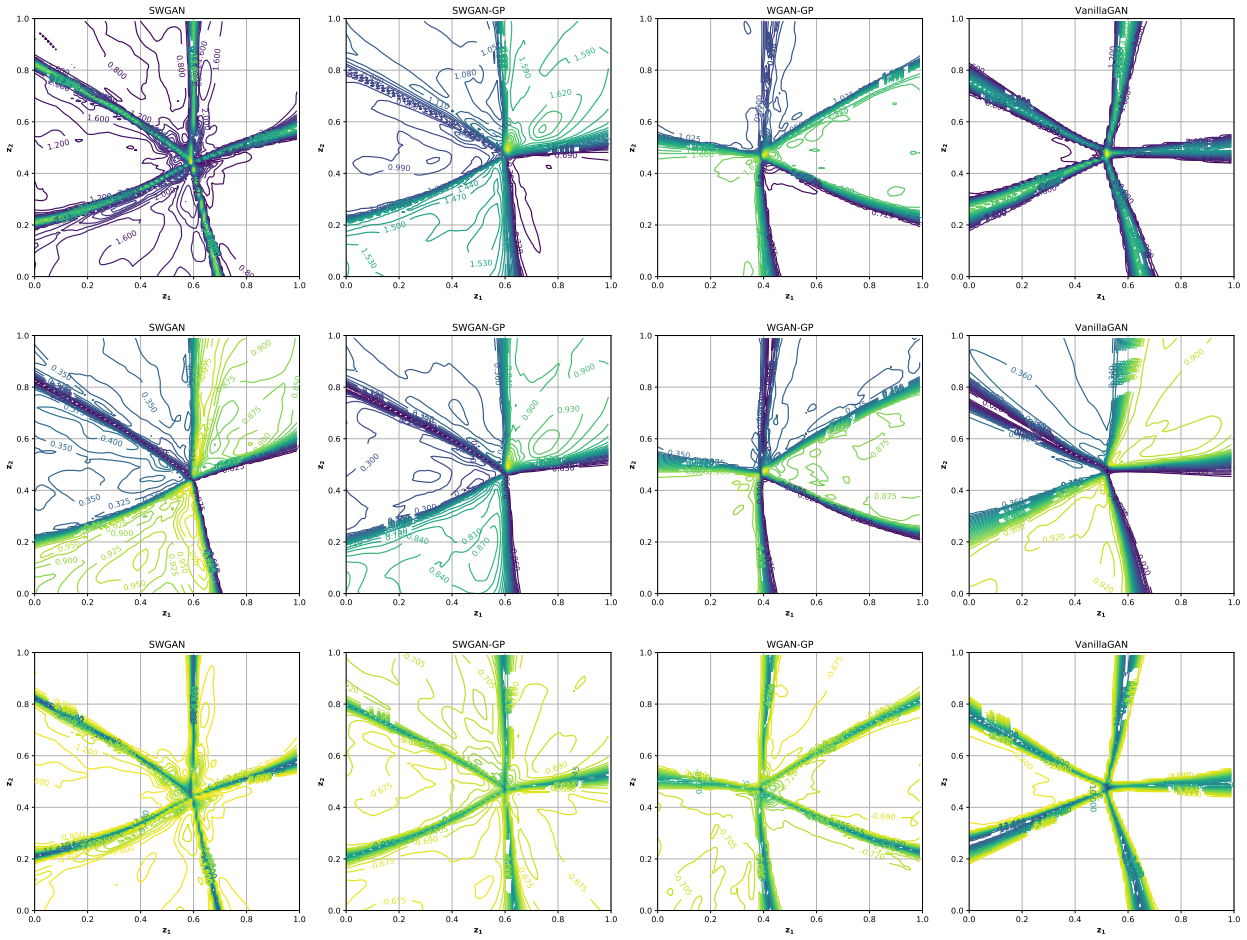


Figure 3.4: Loss in $\mathcal{X}$ space: (a) Vanilla, (b) WGAN-GP, (c) SWGAN, (d) SWGAN-GP at 0, 300, 1200 and 18000 iterations (top to bottom).



(a) (b) (c) (d)

Figure 3.5: Discriminator loss for the synthetic dataset.



(a) (b) (c) (d)

Figure 3.6: Loss surface corresponding to sum of reconstruction and discriminator loss (row 1), reconstruction loss only (row 2), discriminator loss only (row 3) for (a) SWGAN (b) SWGAN-GP (c) WGAN-GP (d) Vanilla GAN at 19000 iterations of training.

erator.

To study more closely whether such raggedness in the optimization landscape is solely caused by the likelihood loss L, we show the loss landscape for the GAN's trained on synthetic data in Fig. 3.6. We note that the optimization landscape is still ragged and the difficulty of the optimization doesn't stem from just the discriminator loss.

As the generator captures the observed data more accurately, we introduce more and more local optima, making it harder and harder to find an accurate result when solving Eq. (3.1). This immediately raises the question: how to address this challenge?

3.2 ROBUST APPROACHES FOR CO-GENERATION

In the following, we provide and analyze three methods to address the question raised in the previous section. We find these mechanisms to be robust across different optimization algorithms.
Method 1: Training Augmentations. Our first method combines three black-box ploys (*i.e.*, no previous knowledge of or access to the generator is needed except for the current model) that directly target the geometry of the loss landscape. We name those ploys 'Augmentation.'

A better initialization is an intuitive way to avoid local optima during co-generation. Towards this goal, by optimizing many randomly initialized $\boldsymbol{z}$'s we hope that some will end up close to the global optimum. However, optimizing multiple $\boldsymbol{z}$'s until convergence is costly. Therefore we trade some accuracy for computational efficiency. We pick as the initializer from a set of sampled $\boldsymbol{z}$'s the one that most accurately generates the exposed portion of the target output image. *I.e.*, we start optimization from

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}\in\hat{\mathcal{Z}}}{\arg\min}\, R(G(\boldsymbol{z})_e, \boldsymbol{x}_e), \tag{3.3}$$

where $\hat{\mathcal{Z}}$ is a set of randomly sampled perturbations. This method is motivated by the observation that the $\mathcal{Z}$-space is often tiled up into plateaus, *i.e.*, 'plains' at different elevation (see bottom row of Fig. 3.3). Hitting a good plateau may be a sufficiently accurate initialization.

We further target the raggedness by smoothing the likelihood loss. Specifically, instead of the original log-likelihood we use the λ-parametrized log likelihood loss as

$$L_\lambda(\boldsymbol{z}) = -\log(\lambda + D_w(G(\boldsymbol{z}))). \tag{3.4}$$

Note that the range of $L_\lambda(\boldsymbol{z})$ is constrained to $(-\log(\lambda+1), -\log(\lambda))$ and ensures that the reconstruction loss R is never dwarfed by the likelihood loss L_λ. In our experiments we employ a constant λ. However, it is also possible to adaptively set $\lambda \propto \frac{1}{k}$, where k is the iteration index.

However, this ploy is not applicable to all co-generation tasks. For example, when the generator

and discriminator is trained using WGAN-GP [37], the likelihood of a generated sample to be true cannot be evaluated using the log-sigmoid of the discriminator's output.

Based on the empirical observation that gradient descent-based optimization often pushes the latent variable $\boldsymbol{z}$ out of the sampling range used during GAN training (when $\boldsymbol{z}$ is sampled from a finite domain such as $\mathcal{U}(0,1)$) or to regions where it is sampled with low probability in training (when $\boldsymbol{z}$ is sampled from an infinite domain such as $\mathcal{N}(0,1)$). The loss landscape outside of the 'confidence zone' is unpredictable. Therefore we enforce a hard or soft constraint on $\boldsymbol{z}$ to avoid this space.

To enforce a hard constraint we clip $\boldsymbol{z}$ to its original domain after each optimization step. For a soft constraint we use a penalty term added to the objective, *i.e.*, we solve

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} R(G(\boldsymbol{z})_e, \boldsymbol{x}_e) + \alpha \cdot L(D(G(\boldsymbol{z}))) - \beta \cdot \log(p(\boldsymbol{z})), \tag{3.5}$$

where $p(\boldsymbol{z})$ is the prior probability of $\boldsymbol{z}$ and β is the weight of the penalty term. Intuitively, we discourage optimization to yield results with a lower probability $p(\boldsymbol{z})$.

In all our experiments we refer to the combination of multiple initial $\boldsymbol{z}$'s, λ-smoothing and clipping (or applying the penalty term) as 'augmentation.' Besides, we also introduce two white-box methods that can be freely combined with the augmentation method.

Method 2: Multi-model Co-generation. As observed in Fig. 3.2 and Fig. 3.3, the loss landscape in $\boldsymbol{z}$-space becomes increasingly ragged as GAN training proceeds. Thus we hypothesize that usage of models from earlier iterations can drive $\boldsymbol{z}$ close to the optimum $\boldsymbol{z}^*$ more easily. Then we can improve the accuracy of optimization by gradually moving on to later models.

To formalize this multi-model method: For a set of generators and discriminators $G_1, \ldots, G_k$ and $D_1, \ldots, D_k$, where G_i and D_i are the models obtained after t_i iterations of GAN training with $t_1 < \ldots < t_k$, we distribute the original number of co-generation update steps n into $n = n_1 + \ldots + n_k$ and optimize $\boldsymbol{z}$ for n_i steps using sequentially for $i = 1, \ldots, k$ the objective:

$$R(G_i(\boldsymbol{z})_e, \boldsymbol{x}_e) + \alpha \cdot L(D_i(G_i(\boldsymbol{z}))). \tag{3.6}$$

The total number of optimization steps stays the same. Note that this method requires access to models from different stages of GAN training, which may not be available. Note, the multi-model method works best if the $\boldsymbol{z}$-space of model i is reasonably close to the $\boldsymbol{z}$-space of model $i+1$ $\forall i$.

Method 3: Layer-wise Co-generation. Raggedness of loss landscape is caused by the complexity of the generator which typically consists of multiple convolutional, normalization and nonlinear layers. It is generally easier to optimize one layer at a time starting from layers closer to the output $\hat{\boldsymbol{x}}$ and moving on to layers closer to the input $\boldsymbol{z}$. Hence, for a generator $G = l_1 \circ l_2 \circ ... \circ l_m$

composed of layers l_i, we optimize from l_1 to l_m using n steps for each:

$$\begin{aligned}
&\arg\min_{\boldsymbol{z}_1} R(l_1(\boldsymbol{z}_1)_e, \boldsymbol{x}_e) + \alpha \cdot L(D(l_1(\boldsymbol{z}_1))) \quad \text{or} \\
&\arg\min_{\boldsymbol{z}_i} \gamma \|l_i(\boldsymbol{z}_i) - \boldsymbol{z}_{i-1}\|_2 + \alpha L(D(l_1 \circ \ldots \circ l_i(\boldsymbol{z}_i))) \\
&+ R(l_1 \circ \ldots \circ l_i(\boldsymbol{z}_i)_e, \boldsymbol{x}_e) \quad \text{for } i > 1,
\end{aligned} \tag{3.7}$$

where $\boldsymbol{z}_i$ is the input vector of layer l_i. That is, for each layer we find the optimal input vector that produces an output as close to the optimal input of the next layer as possible. Importantly, the first term of Eq. (3.7) can be minimized analytically for batch norm [80], linear and leaky ReLU layers. We use this analytical minimum as an initialization for z.

Layer-wise co-generation is also a white-box method as it requires knowledge of the generator architecture and access to individual layers. Moreover, since we need to run the same number of optimization steps for each layer sequentially, layer-wise co-generation is typically slower. For this purpose we add an additional term as in Eq. (3.7) weighted by γ to constraint that $\boldsymbol{z}_i$ produces a similar $\hat{\boldsymbol{x}}$ as any $\boldsymbol{z}_j$, where $i > j$. Throughout we use $\alpha = \gamma = 1$.

In the following we test combinations of the three methods on co-generation tasks using different datasets and GAN models. We investigate the advantages and limitations of each method as well as their interactions.

CHAPTER 4: EXPERIMENTS

We evaluate our proposed methods for robust co-generation. To validate our intuition we first demonstrate results on synthetic data before discussing real data.

Baselines: Baselines for co-generation are optimization from a single latent point using LBFGS, referred to as 'None.' We show results for optimization using LBFGS initialized by picking according to Eq. (3.3) one of 1000 latent points, smoothing the likelihood loss using $\lambda = 0.01$ and adding the prior loss, referred to as 'augmentation.' We also show results for picking the initial latent point using the optimal point of a previous model, referred to as 'multi model,' and the results for applying optimization layer wise, referred to as 'multi layer.' Finally, we conduct ablation studies to show the combination of the proposed methods. Note that we found the combination of 'multi model' and 'multi layer' to not perform well, so we do not include it.

When implementing the multi-model co-generation method, we set $n_1 = \cdots = n_{k-1} = 3$ and $n_k = n - 3(k-1)$. That is, we optimize using each intermediate model for three steps and keep the total number of steps unchanged. $t_1, \ldots, t_k$ are equally spaced with $t_1 = 0$. When combining it with the augmentation method, we calculate the initial $\hat{z}$ using the last model.

When implementing the layer-wise co-generation method, we optimize each layer for n timesteps from the first layer (*i.e.*, the layer immediately before the output) to the last (*i.e.*, the layer immediately following the input). We calculate exact solution of $z_i^* = \arg\min_{z_i} \|l_i(z_i) - z_{i-1}\|_2$ for linear, batch normalization, leaky ReLU layers and use that as the initial z_i for optimization. Note that because our optimization loss includes other terms such as the likelihood loss and the reconstruction loss, we cannot simply declare z_i^* to be the optimal solution.

When combining layer-wise co-generation with the augmentation method, we randomize z_i for each i for multiple times and pick the one that yields the lowest loss.

4.1 SYNTHETIC DATA

The synthetic 2-dimensional data consists of five Gaussians with a variance of 0.02 and means distributed evenly on a circle of radius 1. Our architecture for the generator and the discriminator is a fully connected net with three hidden layers. We illustrate the log likelihood $\log p_w(\boldsymbol{x})$ for synthetic data as estimated by the discriminator for different GAN losses in the Fig. 3.5. As expected from a penalty that was introduced to encourage a Lipschitz constraint, we observe the gradient penalty to increase the smoothness of the $\log(D(\boldsymbol{x}))$ landscape. Intuitively, we expect this smoothness to be beneficial for co-generation.

To assess co-generation we reconstruct $\boldsymbol{x} = (x_1, x_2)$ given $x_2 = 0$. For this task, we show in

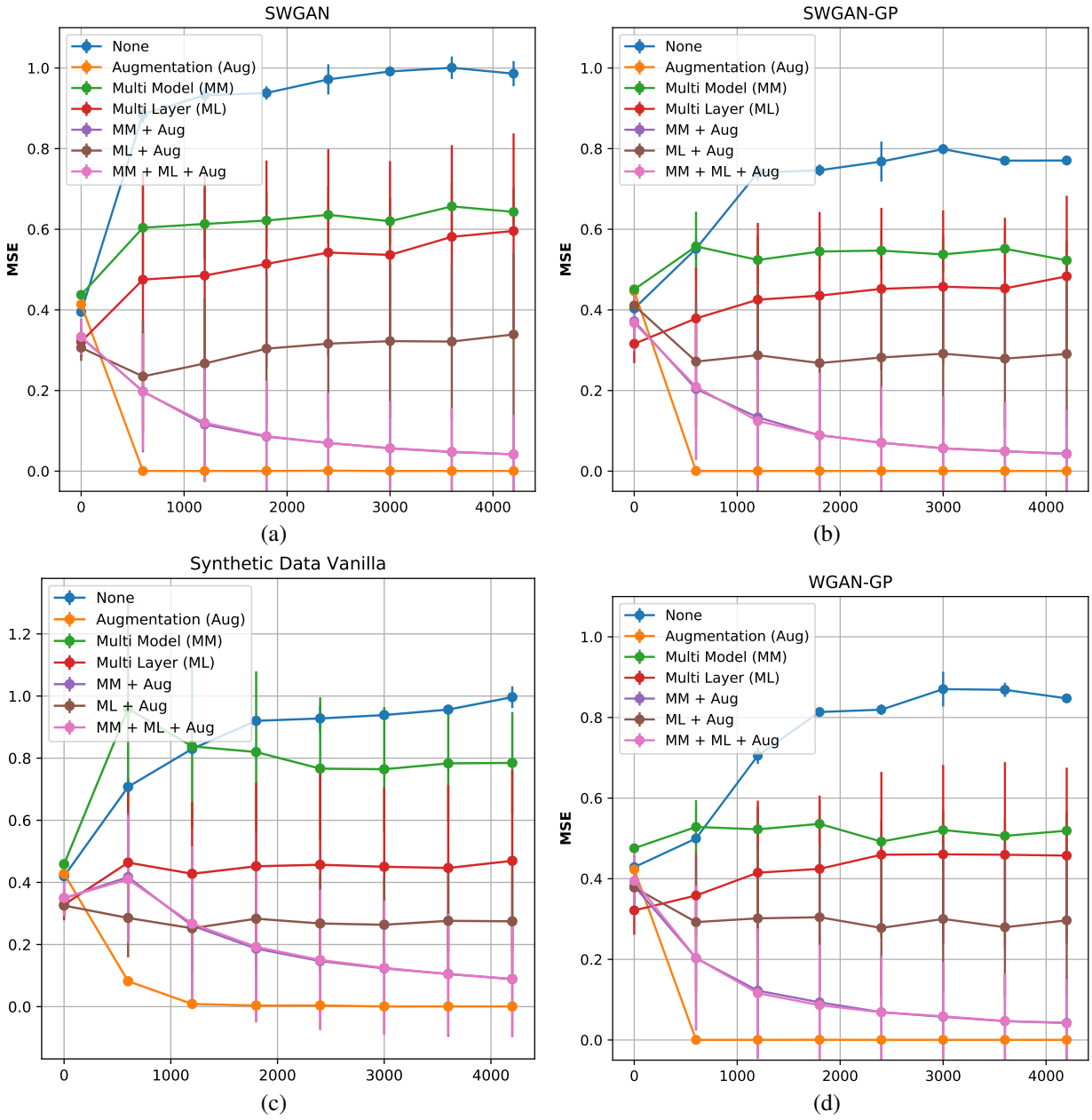


Figure 4.1: Reconstruction error over the number of GAN training iterations for different GAN losses ((a) SWGAN; (b) SWGAN-GP, (c) Vanilla GAN, (d) WGAN-GP) and co-generation approaches (see legend) on synthetic data.

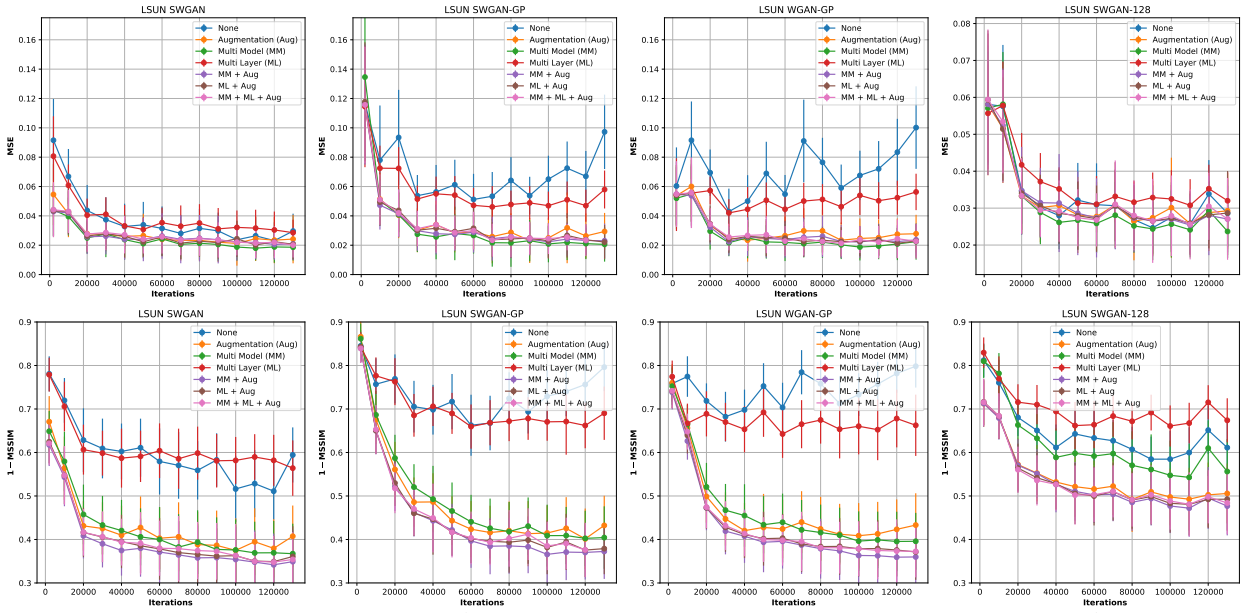


Figure 4.2: Reconstruction error over the number of GAN training iterations for different GAN losses ((a) SWGAN; (b) SWGAN-GP, (c) WGAN-GP, (d) SWGAN (128x128)) and co-generation approaches (see legend) on the LSUN dataset.

Fig. 3.3 and in Fig. 3.4 the energy surface $R_e + \log D_w(G(\boldsymbol{z}^*))$ in the $\mathcal{X}$ and $\mathcal{Z}$ domain respectively. We also illustrate the energy surface of R_e and $\log D_w(G(\boldsymbol{z}^*))$, separately, in Fig. 3.6 to illustrate that the raggedness comes from both the losses and not just one of them. Clearly, when considering the $\mathcal{X}$-domain illustrated in the Fig. 3.4 , the optimal and desired solution is to choose $(x_1, x_2) = (1, 0)$. However, we note that the energy is extremely flat for most parts of $x_2 = 0$ if we don't use the gradient penalty. Moreover it contains very small local bumps which introduce local optima. When using gradient penalty the energy landscape is less flat and local optima are more clearly visible.

Figure 4.3: Reconstructions for different methods on 64x64 LSUN images for a trained SWGAN at 130k iterations.

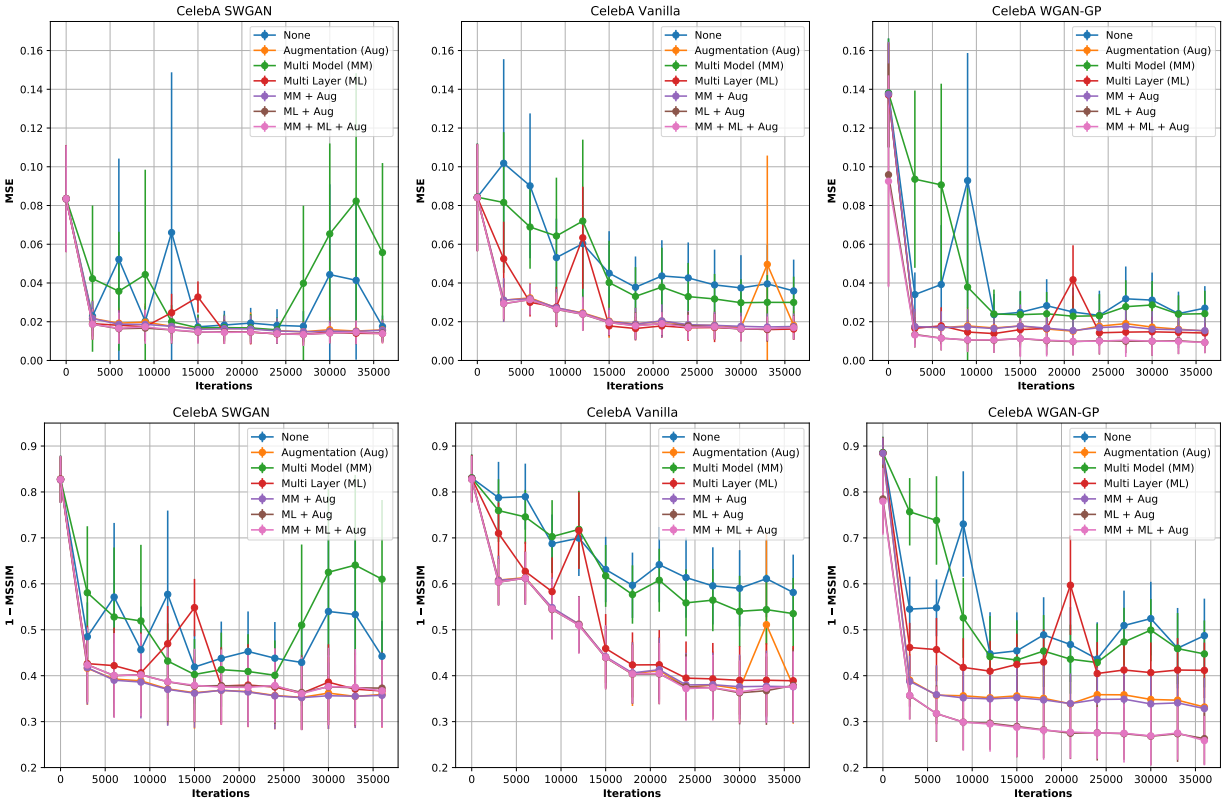


Figure 4.4: Reconstruction error over the number of GAN training iterations for different GAN losses ((a) SWGAN; (b) Vanilla GAN, (c) WGAN-GP and co-generation approaches (see legend) on the CelebA dataset.

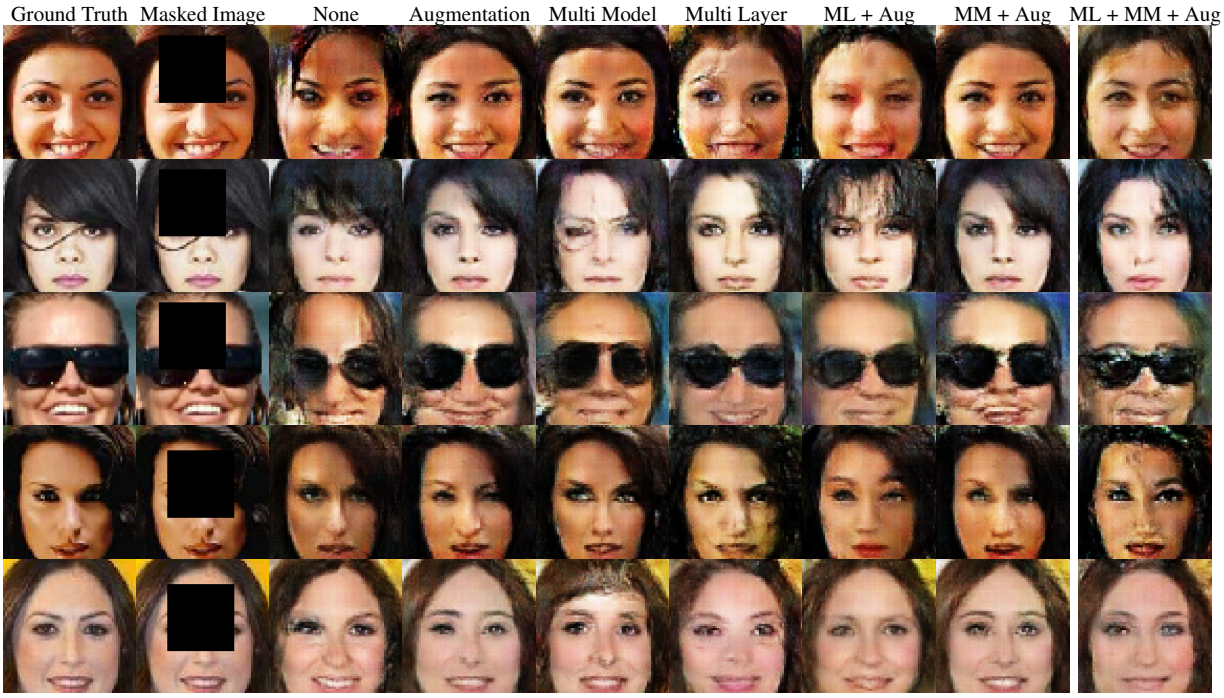


Figure 4.5: Reconstructions for different methods on 64x64 CelebA images for a trained SWGAN at 60000 iterations.

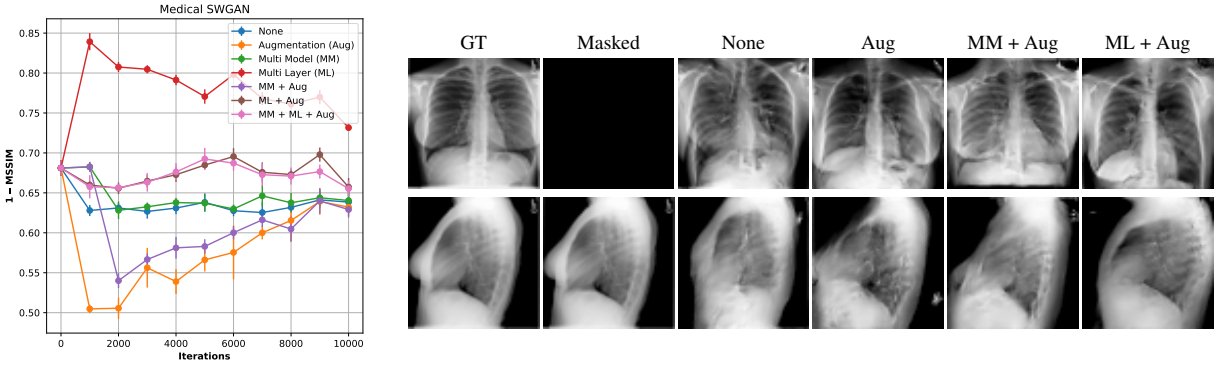


Figure 4.6: Reconstructions for 64x64 medical images with the front view hidden for a trained SWGAN at 10000 epochs. Due to limited space we do not show samples from all methods here.

When considering the $\mathcal{Z}$-domain illustrated in Fig. 3.3, which is important for optimization, we notice huge barriers in the energy landscape particularly when not using gradient penalty. Moreover, we note that the basins are extremely flat, *i.e.*, once initialized to a basin there is hardly any escape which explains the observations illustrated in Fig. 3.2.

To overcome these issues, we proposed a set of simple techniques in Sec. 3.2 which we assess next.

Results: Fig. 4.1 shows the reconstruction error, $\hat{R}$, when using different methods on different GAN architectures. It is apparent that all our proposed methods work better than the baseline. 'Multi layer' and 'multi model' alone perform better than the baseline but when combined with 'augmentation' they perform well. Note that often we find simple augmentation to work very well.

4.2 REAL DATA

To ensure that the proposed methods are also applicable to real data, we evaluate our approach on three datasets, using MSE and MSSIM as metrics.

LSUN: On the LSUN dataset we train SWGAN, SWGAN-GP and WGAN-GP on images of size 64x64, and SWGAN on images of size 128x128, using a DCGAN [81] architecture for the generator and discriminator (we replace BatchNorms with LayerNorms in both since it improved co-generation). To assess co-generation, we mask out a randomly chosen block of size one-fourth ($\boldsymbol{x}_e$) and reconstruct the original image using the program given in Eq. (3.1).

In Fig. 3.2 (b) we show that the reconstruction error $\hat{R}$ doesn't improve as GAN training progresses. In Fig. 4.2 we observe that using the proposed solutions indeed helps. Using 'augmentation,' co-generation remains stable as training progresses irrespective of the GAN training loss. Although 'multi model' gives a lower MSE than 'augmentation', we see in Fig. 4.2 (d) that MSSIM suffers. However, combining 'augmentation' with 'multi model' improves MSSIM. Also, 'multi layer' doesn't work as well with GANs at a resolution of 128×128 as shown in Fig. 4.2 (d). This is likely due to larger number of layers and accumulating reconstruction errors. Fig. A.6 illustrates generated results and shows that the methods combined with 'augmentation' are robust to the location of the mask.

CelebA: On the CelebA dataset we train SWGAN, Vanilla GAN and WGAN-GP for images of size 64x64. The network architecture and masking rule for co-generation are identical to the LSUN experiment.

The comparison across different co-generation methods of the three GANs is illustrated in Fig. 4.4. Again we see that 'augmentation' works well and when added to the 'multi model' and 'multi layer' approaches keep co-generation stable and improves in some cases. Fig. A.7

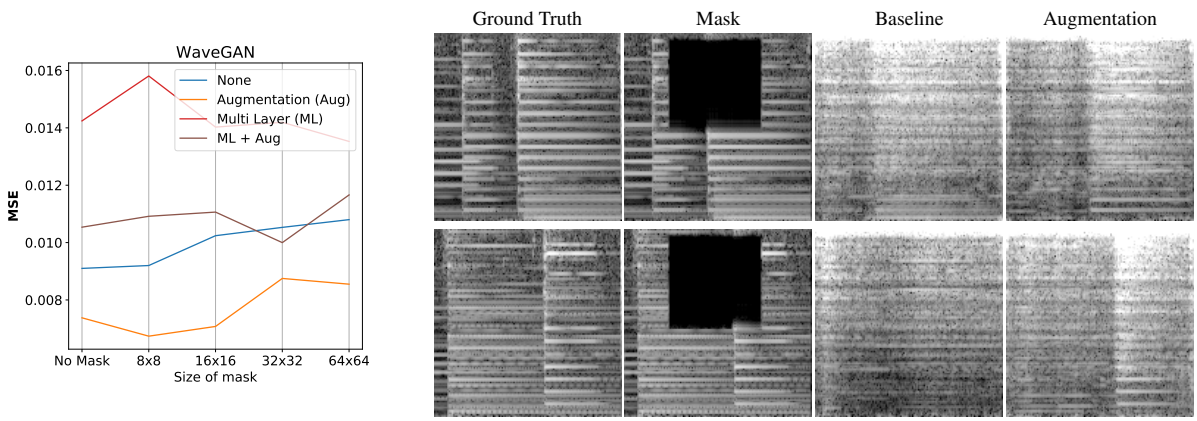


Figure 4.7: Cogeneration on WaveGAN.

illustrates generated results showing that our method is robust to location of the missing part.

Medical Data: The medical dataset we use consists of 4624 pairs of anteriorposterior and lateral chest X-ray images. The original dataset is rescaled to 64x64 and histogram-equalized before training.

The generator has two streams that take the same $\boldsymbol{z}$ as input and each outputs a 1x64x64 image for front and side view. Each stream employs a ResNet [82] architecture with four UpBlocks that contains one transpose convolution layer and two convolution layers with skip connection. The discriminator is built similarly. We were only able to train the SWGAN on this dataset.

For co-generation we use the side view as the exposed portion $\boldsymbol{x}_e$ and the front view as the hidden portion $\boldsymbol{x}_h$. This is a harder task compared to co-generation for LSUN and CelebA because half of the output space is masked. Still our method is able to capture the connection between two views, as demonstrated in Fig. 4.6. In Fig. 4.6 we also compare different methods of co-generation.

We also think that multi layer doesn't work well because of the ReLU layers in the net. The number of optimal solutions for a ReLU is infinite, which wasn't the case earlier because leaky ReLUs were used, giving an analytically computable optimum to initialize the program in Eq. (3.7).

WaveGAN: To illustrate our method beyond inpainting, we show examples to reconstructe audio waves with WaveGAN [83]. We use a trained WaveGAN that generates fixed length piano sounds. Given an audio wave we apply a mask on its spectrogram and transform it back into a wave. We then use this wave as the input to cogeneration and report in Fig. 4.7 the MSE between the actual wave and the cogenerated wave. We notice that 'augmentation' performs best. The resulting spectrograms, however, look different because the model tends to create waves with higher frequencies. We believe that this explains why not using a mask has only a slightly lower MSE than using a mask. Similarly multi layer doesn't work well in this experiment because of the ReLU layers in the network.

CelebA-HQ: Another application of co-generation is single image super-resolution (SISR). To test this application we use the network proposed in [84] (denoted as PGAN), which is trained on the CelebA-HQ dataset to output images of size 1024x1024. For co-generation we take CelebA-HQ images and downsample them to 64x64. Here $\boldsymbol{x}_e$ represents low-resolution images, and therefore the corresponding reconstruction loss is:

$$R(G(\boldsymbol{z})_e, \boldsymbol{x}_e) = \|P(G(\boldsymbol{z})) - P(\boldsymbol{x})\|_2, \tag{4.1}$$

where P denotes downsampling via average pooling. We evaluate the performance of our co-generation methods via MSE and MSSIM between the final output and the high-resolution groundtruth.

We don't focus on comparing different combinations of methods since we only have access to the final PGAN model and layer-wise co-generation on such a huge network is memory demanding. Comparing the baseline and augmentation gives an average MSE of 0.4831 and 0.0248 respectively and an average 1 - MSSIM of 0.9198 and 0.3571 respectively, demonstrating the benefits of the proposed approach. Some example images are shown in Fig. 4.8.

Figure 4.8: SISR for CelebA-HQ images from 64x64 to 1024x1024 using PGAN.

CHAPTER 5: CONCLUSION

We analyzed co-generation, *i.e.*, the task of obtaining a sample for a subset of the output space domain given as input the remainder. For the first time, some challenges that classical approaches face when reconstructing partially observed data by back-propagating to the latent space are identified and illustrated. To alleviate those challenges, we developed three methods for co-generation with GANs based on obtained insights. On four datasets we show that the three techniques improve reconstruction. We hope that the provided analysis spurs future research for co-generation, a challenging yet important problem.

APPENDIX A: APPENDIX

In this appendix we provide some additional analysis (see Sec. A.1) and qualitative results (see Sec. A.2).

A.1 ADDITIONAL ANALYSIS

Fig. A.1 also shows another method discussed in [77] to initialize Z. This method basically tries to predict the initial Z using another neural network. We did not find this to work well in practice.

In Fig. A.2 we show the effect of using a discriminator *vs*. not using a discriminator on different GANs trained on the LSUN dataset when using the 'augmentation method.' While introducing the discriminator loss slightly increases the MSE between the reconstructed image and the ground truth from the LSUN dataset, it decreases the other reconstruction error metric, *i.e*., 1-MSSIM, and it improves the overall diversity of the reconstructed image space evaluated by the FID score [35]. Since the difference in MSE is not too significant when using the discriminator loss but we have evidence that using it improves MSSIM and FID scores in general, we choose to use it.

A.2 ADDITIONAL QUALITATIVE RESULTS

To illustrate the cogeneration process when using a multi layer approach we show the reconstruction output at each intermediate layer/step in Fig. A.3. We believe there is a major issue for cogeneration in the first linear layer because the dimension goes from 1024 to 1024x1024 which is inherently hard to optimize. We believe this can be circumvented by having multiple linear layers instead of just one.

To illustrate the cogeneration process when using multiple models, we show the output at each intermediate model up until the model at 130k iterations in Fig. A.4 and Fig. A.5. These results show that initially the reconstructions are not detailed which can be attributed to the model. Note

(a) (b) (c) (d) (e)

Figure A.1: (a) true image; (b) masked image; (c) generated image w/o D-Loss; (d) generated image w/ D-Loss; (e) using [77] to reconstruct.

that we get a reasonable approximation to the Ground Truth (GT) pretty early and the later models add to the detail and regularity of the reconstruction.

We show additional qualitative results for different co-generation mechanisms on the 64×64 LSUN dataset in Fig. A.6 and on the 64×64 CelebA dataset in Fig. A.7 when using the SWGAN. We observe our method to accurately recover many of the missing regions. In these figures we compare using Adam/LBFGS as optimizers, one initial Z *vs*. multiple initial Z's (10kZ), using λ *vs*. not using λ and using clipping with LBFGS, 10kZs and lambda (our method for augmentation). Note that different missing regions for the same image result in similarly recovered scenes or faces for the proposed technique, illustrating the stability.

We also ran our experiment on datasets with higher resolution. In Fig. A.8, Fig. A.9 and Fig. A.10 we compare the co-generation task using our method and raw LBFGS optimization with a single initial $\boldsymbol{z}$ on 256×256 CelebA-HQ data. We see that our method works better on approximating the real images even when one quarter of the real images is masked.

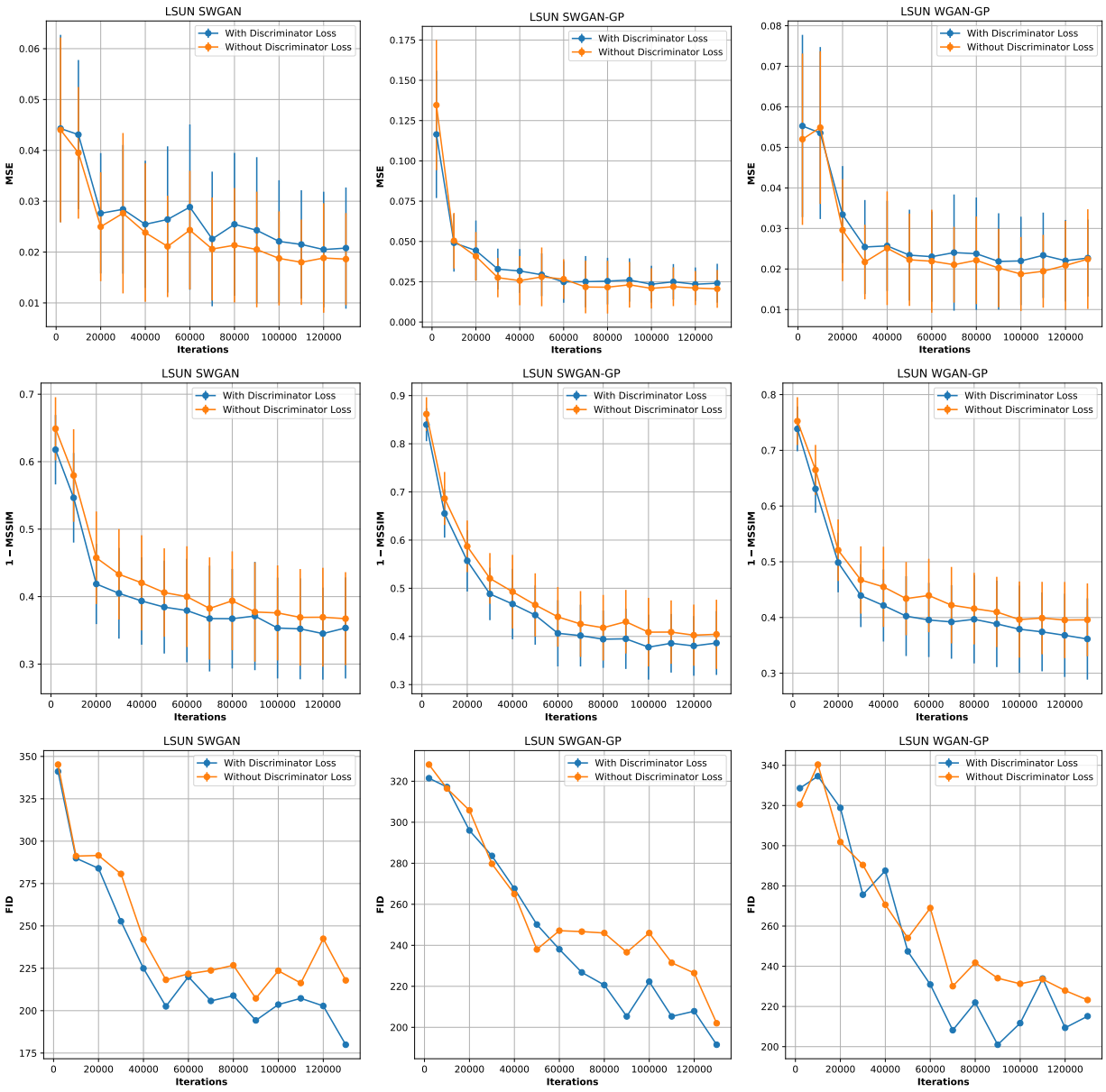


Figure A.2: Effect of including and excluding the discriminator loss in the loss function on (a) SWGAN; (b) SWGAN-GP, (c) WGAN-GP.

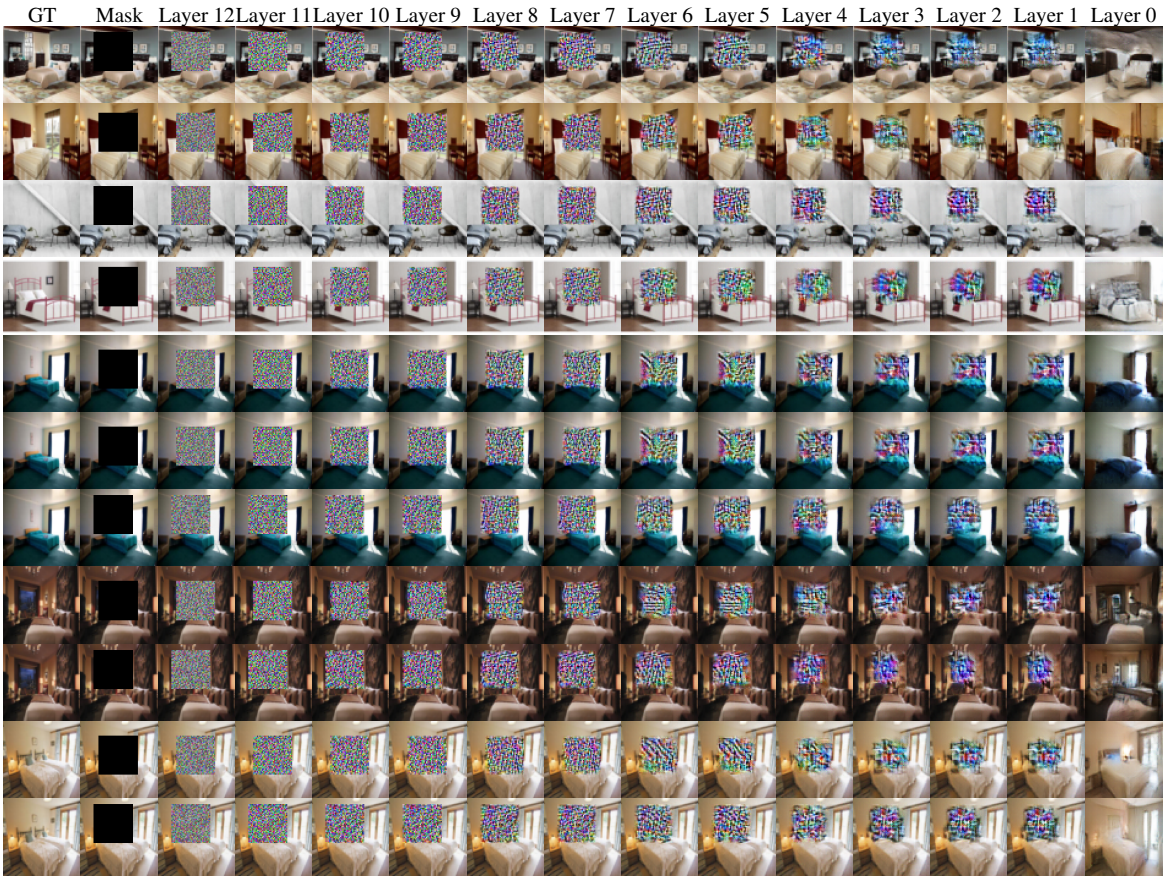


Figure A.3: Reconstruction at each layer.

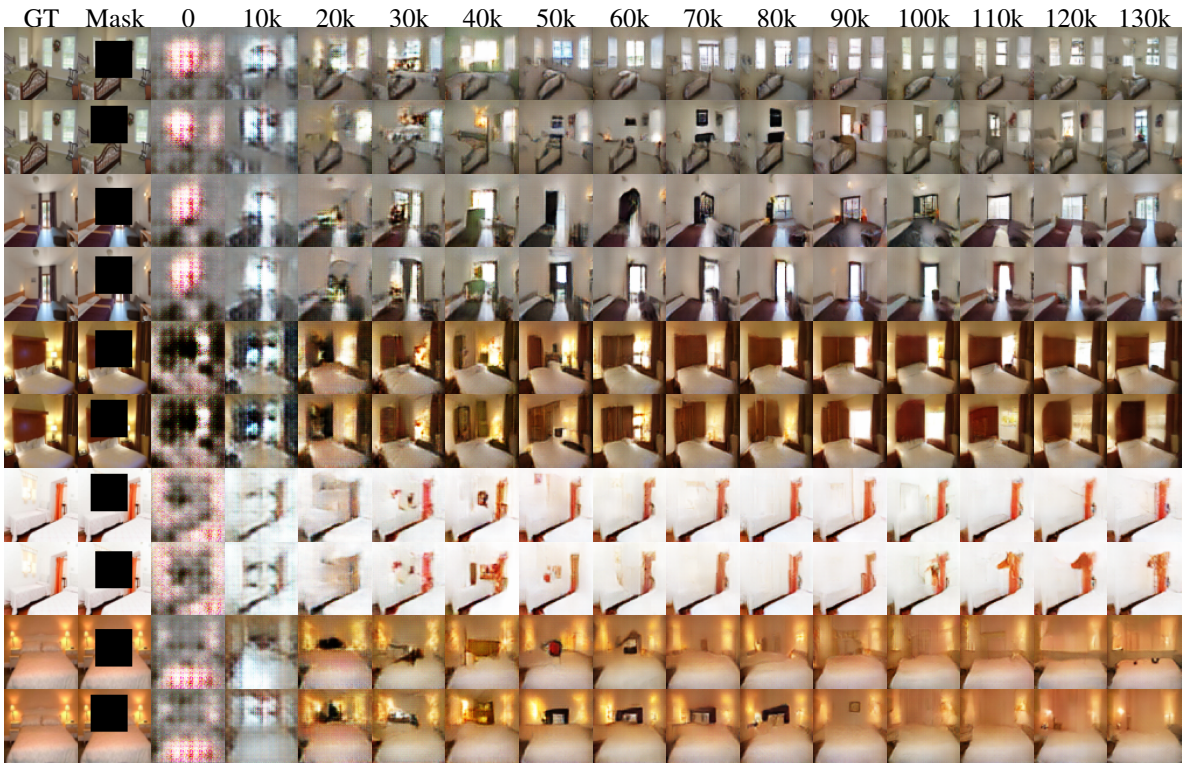


Figure A.4: Reconstruction for different models on LSUN.

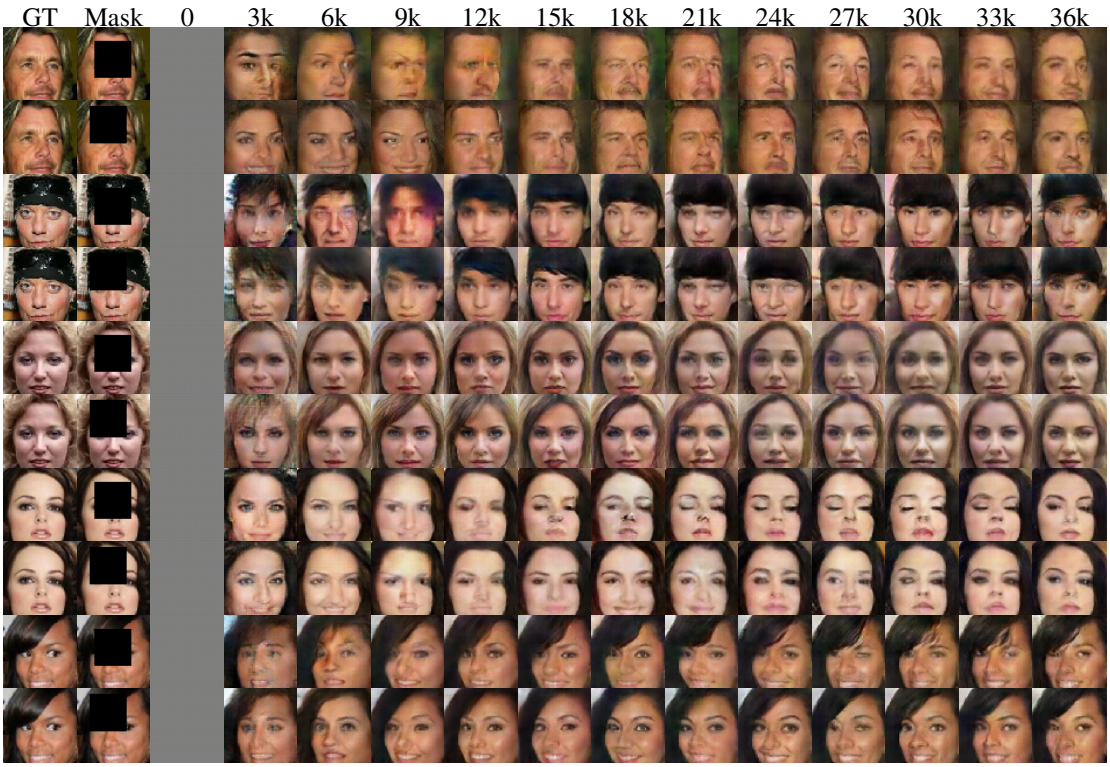


Figure A.5: Reconstruction for different models on CelebA.

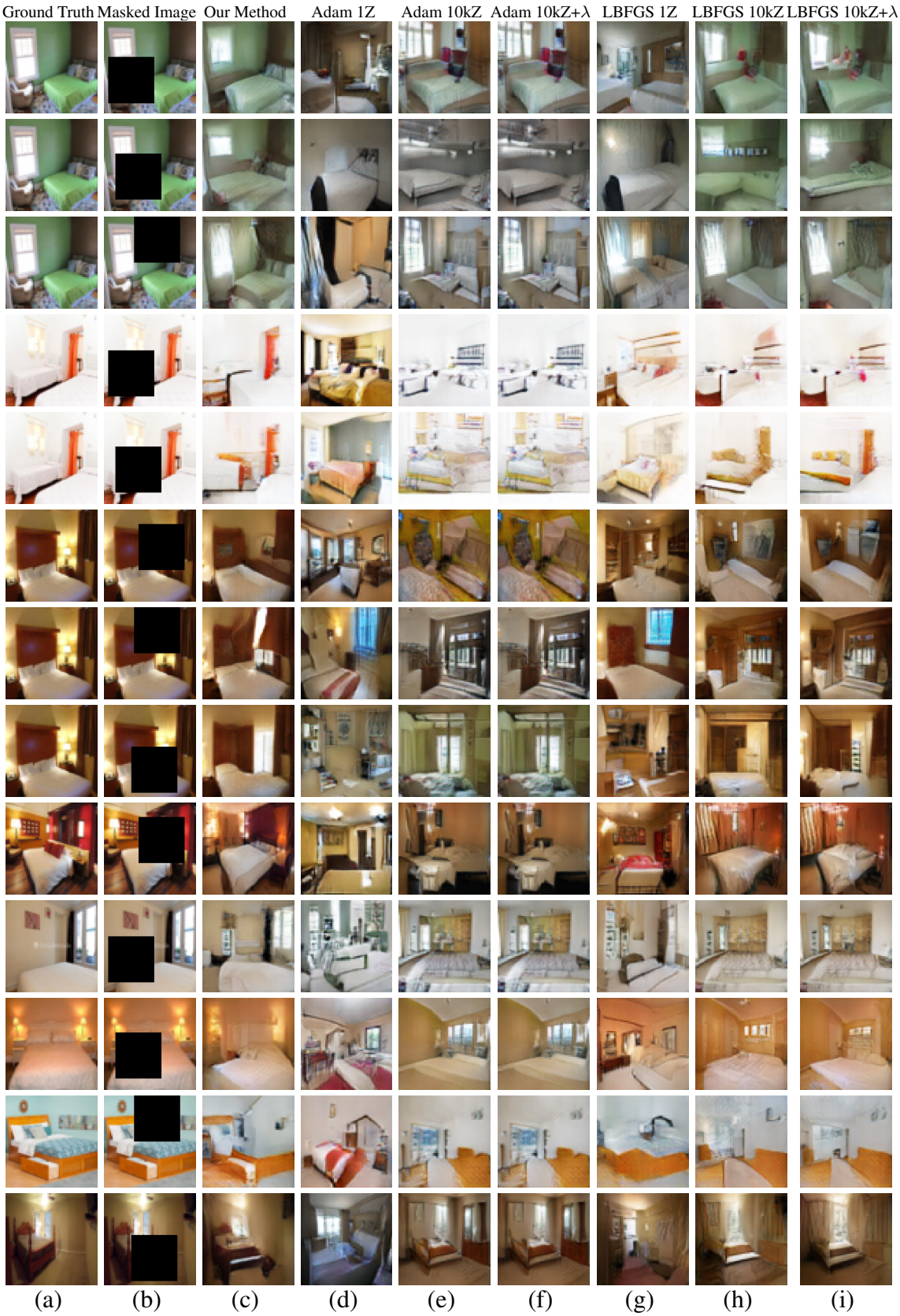


Figure A.6: Reconstructions for different methods on 64x64 LSUN images for a trained SWGAN at 120000 iterations.

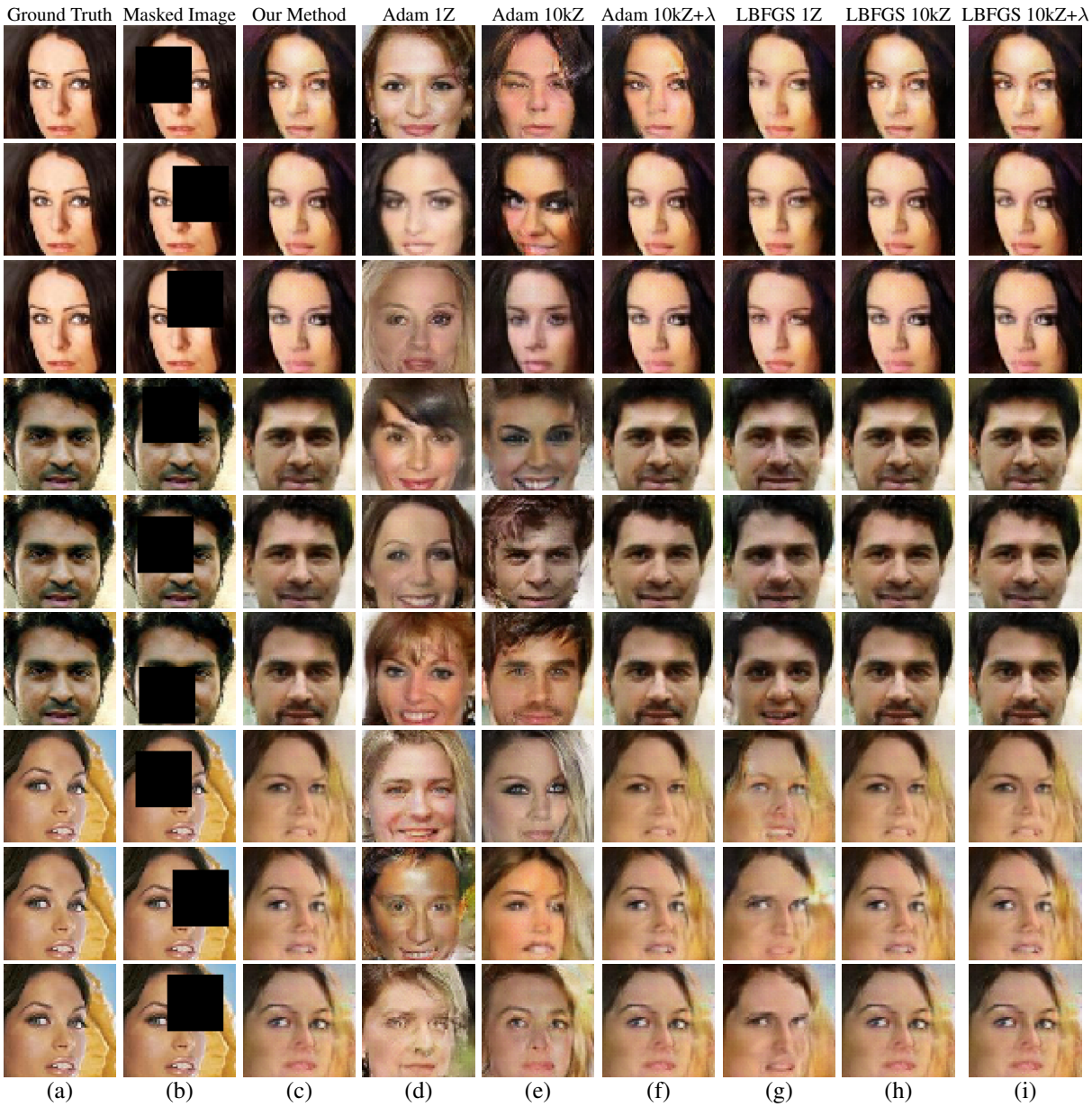


Figure A.7: Reconstructions for different methods on 64x64 CelebA images for a trained SWGAN at 60000 iterations.

Figure A.8: Downsampled CelebA-HQ data.

Figure A.9: Co-generated high-resolution CelebA-HQ images with baseline method.

Figure A.10: Co-generated high-resolution CelebA-HQ images with augmentation method.

REFERENCES

[1] A. Anoosheh, E. Agustsson, R. Timofte, and L. Van Gool, "Combogan: Unrestrained scalability for image domain translation," in *arXiv:1712.06909*, 2017.

[2] Y. Choi, M. Choi, M. Kim, J. W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proc. CVPR*, 2018.

[3] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. CVPR*, 2016.

[4] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic Image Inpainting with Deep Generative Models," in *Proc. CVPR*, 2017.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. CVPR*, 2017.

[6] H. Y. Lee, H. Y. Tseng, J. B. Huang, M. K. Singh, and M. H. Yang, "Diverse image-to-image translation via disentangled representation," in *Proc. ECCV*, 2018.

[7] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal Unsupervised Image-to-Image Translation," in *Proc. ECCV*, 2018.

[8] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks," in *Proc. NIPS*, 2017.

[9] A. Royer, K. Bousmalis, S. Gouws, F. Bertsch, I. Moressi, F. Cole, and K. Murphy, "Xgan: Unsupervised image-to-image translation for many-to-many mappings," in *arXiv:1711.05139*, 2017.

[10] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proc. ICCV*, 2017.

[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. ICCV*, 2017.

[12] J. Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Proc. NIPS*, 2017.

[13] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," *arXiv preprint arXiv:1710.04087*, 2017.

[14] H.-C. Shin, N. A. Tenenholtz, J. K. Rogers, C. G. Schwarz, M. L. Senjem, J. L. Gunter, K. Andriole, and M. Michalski, "Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks," in *https://arxiv.org/abs/1807.10225*, 2018.

[15] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for segmenting and labeling sequence data," in *Proc. ICML*, 2001.

[16] B. Taskar, C. Guestrin, and D. Koller, "Max-Margin Markov Networks," in *Proc. NIPS*, 2003.

[17] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," *JMLR*, 2005.

[18] H. Rue, *Gaussian markov random fields: theory and applications*. CRC Press, 2008.

[19] S. E. Shimony, "Finding MAPs for belief networks is NP-hard," *Artificial Intelligence*, 1994.

[20] L. G. Valliant, "The complexity of computing the permanent," *Theoretical Computer Science*, 1979.

[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," in *https://arxiv.org/abs/1406.2661*, 2014.

[22] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *https://arxiv.org/abs/1312.6114*, 2013.

[23] M.-Y. Liu and O. Tuzel, "Coupled Generative Adversarial Networks," in *Proc. NIPS*, 2016.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014.

[25] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[26] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," in *Proc. NIPS*, 2017.

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[28] S. Kolouri, G. K. Rohde, and H. Hoffman, "Sliced wasserstein distance for learning gaussian mixture models," *arXiv preprint arXiv:1711.05376*, 2017.

[29] I. Deshpande, Z. Zhang, and A. Schwing, "Generative modeling using the sliced wasserstein distance," *arXiv preprint arXiv:1803.11188*, 2018.

[30] R. W. A. Cully, H. J. Chang, and Y. Demiris, "Magan: Margin adaptation for generative adversarial networks," *arXiv preprint arXiv:1704.03817*, 2017.

[31] Y. Mroueh, T. Sercu, and V. Goel, "Mcgan: Mean and covariance feature matching gan," *arXiv preprint arXiv:1702.08398*, 2017.

[32] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[33] Y. Mroueh and T. Sercu, "Fisher GAN," in *Proc. NIPS*, 2017.

[34] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," *arXiv preprint arXiv:1712.04086*, 2017.

[35] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. NIPS*, 2017.

[36] T. Salimans, H. Zhang, A. Radford, and D. Metaxas, "Improving gans using optimal transport," *arXiv preprint arXiv:1803.05573*, 2018.

[37] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," in *Proc. NIPS*, 2017.

[38] M. Mirza and S. Osindero, "Conditional generative adversarial nets," in *arXiv preprint arXiv:1411.1784*, 2014.

[39] T. C. Wang, M. Y. Liu, J. Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Proc. CVPR*, 2018.

[40] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep Boltzmann machines," in *Proc. NIPS*, 2012.

[41] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proc. ICML*, 2011.

[42] R. Kiros, R. R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models," in *arXiv:1411.2539*, 2014.

[43] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim, "Rotating your face using multi-task deep neural network," in *Proc. CVPR*, 2015.

[44] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Proc. NIPS*, 2015.

[45] A. Dosovitskiy, J. T. Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *Proc. CVPR*, 2015.

[46] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. ECCV*, 2016.

[47] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. CVPR*, 2017.

[48] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proc. ICCV*, 2017.

[49] X. Liang, H. Zhang, and E. P. Xing, "Generative semantic manipulation with contrasting GAN," in *arXiv:1708.00315*, 2017.

[50] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," in *Proc. ICLR*, 2017.

[51] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proc. CVPR*, 2017.

[52] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proc CVPR*, 2017.

[53] L. Wolf, Y. Taigman, and A. Polyak, "Unsupervised creation of parameterized avatars," in *Proc. ICCV*, 2017.

[54] T. G. Tau, L. Wolf, and S. B. Tau, "The role of minimal complexity functions in unsupervised learning of semantic mappings," in *Proc. ICLR*, 2018.

[55] Y. Hoshen and L. Wolf, "Identifying analogies across domains," in *Proc. ICLR*, 2018.

[56] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proc. ICML*, 2017.

[57] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville, "Augmented Cycle-GAN: Learning many-to-many mappings from unpaired data," in *arXiv:1802.10151*, 2018.

[58] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," in *Proc. NIPS*, 2017.

[59] Z. Gan, L. Chen, W. Wang, Y. Pu, Y. Zhang, H. Liu, C. Li, and L. Carin, "Triangle generative adversarial networks," in *Proc. NIPS*, 2017.

[60] C. Li, K. Xu, J. Zhu, and B. Zhang, "Triple generative adversarial nets," in *Proc. NIPS*, 2017.

[61] P. Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays, "Transient attributes for high-level understanding and editing of outdoor scenes," *TOG*, 2014.

[62] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proc. SIGGRAPH*, 2001.

[63] J. B. Tenenbaum and W. T. Freeman, "Separating style and content," in *Proc. NIPS*, 1997.

[64] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Proc. NIPS*, 2016.

[65] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, "Decomposing motion and content for natural video sequence prediction," in *Proc. ICLR*, 2017.

[66] E. Denton and V. Birodkar, "Unsupervised learning of disentangled representations from video," in *Proc. NIPS*, 2017.

[67] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun, "Disentangling factors of variation in deep representation using adversarial training," in *Proc. NIPS*, 2016.

[68] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," in *Proc. ICML*, 2016.

[69] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. ICCV*, 2017.

[70] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. H. Yang, "Universal style transfer via feature transforms," in *Proc. NIPS*, 2017.

[71] Y. Li, M. Y. Liu, X. Li, M. H. Yang, and J. Kautz, "A closed-form solution to photo-realistic image stylization," in *Proc. ECCV*, 2018.

[72] S. Tulyakov, M. Y. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," in *Proc. CVPR*, 2018.

[73] C. Donahue, A. Balsubramani, J. McAuley, and Z. C. Lipton, "Semantically decomposing the latent spaces of generative adversarial networks," in *Proc. ICLR*, 2018.

[74] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," in *Proc. NIPS*, 2017.

[75] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *Proc. ICLR*, 2017.

[76] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," in *Proc. BMVC*, 2017.

[77] J. Y. Zhu, P. Krähenbühl, E. Shechtman, and A. Efros, "Generative visual manipulation on the natural image manifold," in *Proc. ECCV*, 2016.

[78] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. CVPR*, 2016.

[79] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *TIP*, 2004.

[80] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[81] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *https://arxiv.org/abs/1511.06434*, 2016.

[82] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. CVPR*, 2016.

[83] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *ICLR*, 2019.

[84] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *Proc. ICLR*, 2018.